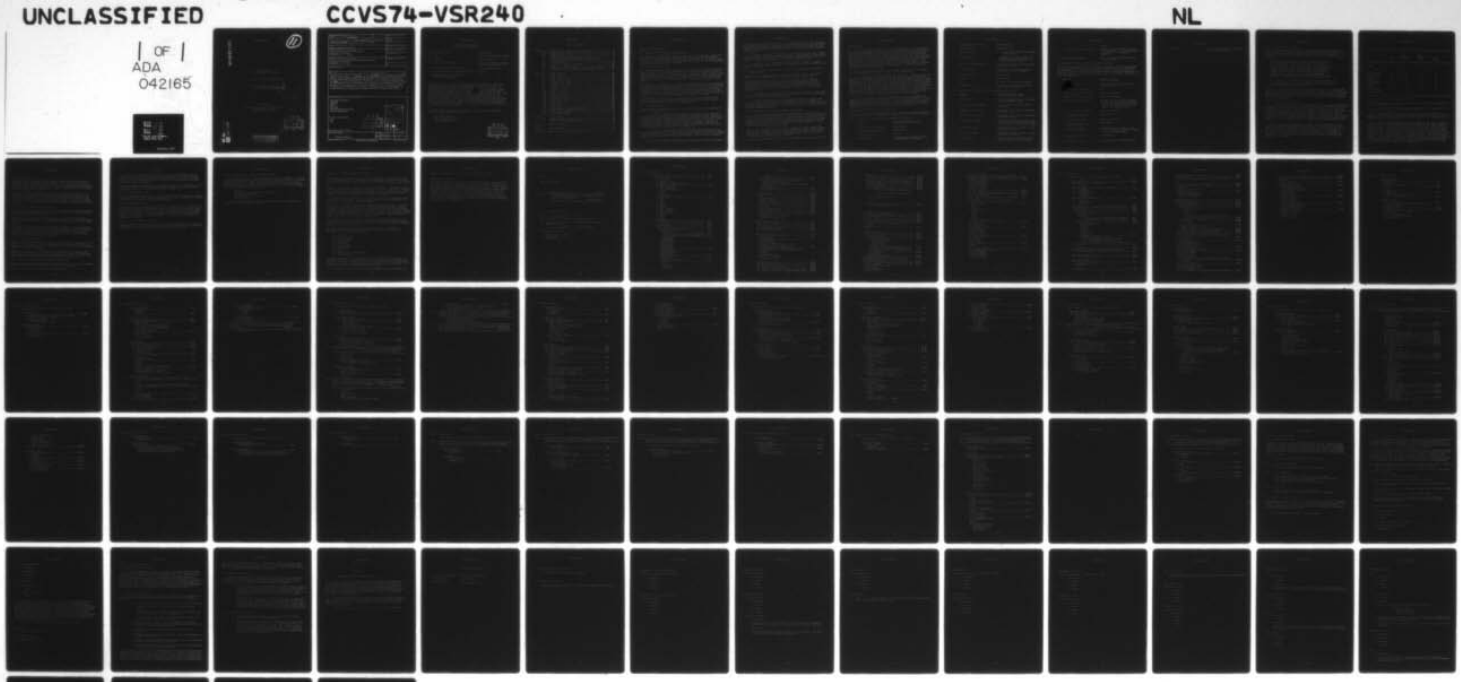
AD-A042 165 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT.(U)
JUL 77

UNCLASSIFIED CCVS74-VSR240 NL

1 OF 1
ADA
042165



END

DATE
FILMED
8-77

CCVS74-VSR240

COBOL COMPILER VALIDATION SUMMARY REPORT

VALIDATION NUMBER CCVS74-VSR240

1 Jul 77

71 p.

Prepared By:

FEDERAL COBOL COMPILER TESTING SERVICE
DEPARTMENT OF THE NAVY
WASHINGTON, D.C. 20376

408 438

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS74-VSR240	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report # CCVS74-VSR240 (Assigned by Manager of Testing) H6180 MULTICS COBOL			5. Report Date 1JUL77 6.
7. Author(s) Same as organization - see 9.			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Federal COBOL Compiler Testing Service Department of the Navy Washington, D. C. 20376			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address Automatic Data Processing Equipment Selection Office Department of the Navy Washington, D. C. 20376			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

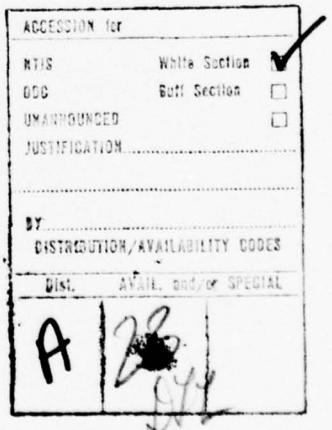
This Validation Summary Report (VSR) for the Honeywell Multics COBOL Compiler Version 2.3 (Multics Version MR4.0) provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1974 COBOL Standard (X3.23-1974/FIPS PUB 21-1). The compiler was validated at the High level of FIPS PUB 21-1. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by level/module within X3.23-1974, a section relating the categories of discrepancies to each of the Federal levels of the language; and a detailed listing of discrepancies together with the tests which were failed.

17. Key Words and Document Analysis. 17a. Descriptors

Porgramming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

17b. Identifiers/Open-Ended Terms

CCVS
CVS



17c. COSATI Field/Group 09/02

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release unlimited.	UNCLASSIFIED	72
	20. Security Class (This Page) UNCLASSIFIED	22. Price

CCVS74-VSR240

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS74-VSR240
2.	Vendor	Honeywell Information Systems
3.	Mainframe	H6180 (Multics)
4.	Compiler Identification	Honeywell Information Systems Multics COBOL Version 2.3
5.	Operating System Identification	Multics Release MR4.0
6.	Compiler Validation System Version Number	CCVS74 2.0
7.	Federal Information Processing Standard Publication	21-1

*PLEASE NOTE. The Federal COBOL Compiler Testing Service may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of this validation are only for the purpose of satisfying United States Government requirements, and apply only to the Computer System, Operating System release, and compiler version identified in the VSR. The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the Federal COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

For information concerning this compiler you can contact the vendor's designated representative named below:

Dr. Clair Miller
HISI, Honeywell Center
7900 Westpark Drive
McLean, Virginia 22101

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1 Purpose of the Validation Summary Report

The purpose of the Validation Summary Report (VSR) is to identify individual COBOL language elements whose implementation does not conform to American National Standard Programming Language COBOL, X3.23-1974, and to Federal Standard COBOL as adopted from the American National Standard by Federal Information Processing Standard 21-1 (FIPS PUB 21-1).

1.2 Preparation of the VSR

The Validation Summary Report is prepared by analyzing the results of running the COBOL Compiler Validation System (CCVS). The COBOL Compiler Validation System consists of audit routines containing features of Federal Standard COBOL, their related data, and an executive routine (VP-routine) which prepares the audit routines for compilation. Each audit routine is a COBOL program which includes many tests and supporting procedures indicating the result of the tests.

The testing of a compiler in a particular hardware/operating system environment is accomplished by compiling and executing each audit routine. The report produced by each routine tells whether the compiler passed or failed the tests in the routine. If the compiler rejects some language elements by terminating compilation, giving fatal diagnostic messages, or terminating execution abnormally, then the test containing the code the compiler was unable to process is deleted and the audit routine compilation and execution repeated.

The compilation listings and the output reports of the audit routines constitute the raw data from which the members of the Federal COBOL Compiler Testing Service produce a Validation Summary Report.

1.3 Organization of the VSR

The Validation Summary Report is made up of several sections the contents of which are described below.

a. Section 2 summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System. Section 2 is subdivided into a subsection representing each level of each module defined in American National Standard Programming Language COBOL, X3.23-1974. Each subsection contains a list of all of the language elements which must be implemented in order to claim support of that level/module. The list of language elements will be annotated to include a description of both syntax and semantic errors detected during the validation.

b. Section 3 - FIPS PUB 21-1 defines four Federal levels of the COBOL Standard. Section 3.1 of the VSR lists the discrepancies described in Section 2 by the Federal level in which the problem occurs. Section 3.2 lists discrepancies for the Report Writer Module, which is not a part of Federal Standard COBOL.

c. Section 4 contains information which describes the software environ-

ment in which the compiler was tested. This includes the name and version of the operating system; the implementor-names which were used in the Environment Division of the programs comprising the CCVS; the options used with the compiler; and if applicable, information regarding the use of compiler optimization features.

d. Section 5 contains the results of the ASCII validation. The purpose of these tests is to ascertain whether magnetic tapes written in ASCII code and with ANSI standard labels, and card decks with ASCII code, can be transported between the system being validated and a foreign computer system.

e. Appendix A is the Validation Summary Working Document, a working paper resulting from the compilation and execution of the CCVS, and from which the VSR is derived.

1.4 Abstract Covering Compliance to American National Standard Programming Language COBOL

Definition of an Implementation of American National Standard Programming Language COBOL (excerpts from X3.23-1974, Chapter 1, Section 1.5).

An implementation is defined to meet the requirements of the American National Standard COBOL specification if that implementation includes a fully implemented specified level of each of the functional processing modules and of the Nucleus as defined in this Standard. It follows from this that, in order to meet the requirements of this Standard, an implementation must:

a. Not require the inclusion of substitute or additional language elements in the source program, in order to accomplish any part of the function of any of the standard language elements.

b. Accept all standard language elements contained in a given level of a module which is specified as being included in the implementation, except as specifically exempted (as pertaining to specific hardware components for which support is not claimed). See "Elements that Pertain to Specific Hardware Components" below.

These points are of particular pertinence in two areas:

(1) There are throughout the American National Standard COBOL specification certain language elements whose syntax, or effect, is specified to be, in part, implementor-defined. While the implementor specifies the constraints on that portion of each element's syntax or rules that is indicated in this Standard to be implementor-defined, such constraints may not include any requirement for the inclusion in the source program of substitute or additional language elements.

(2) When a function is provided outside the source program that accomplishes a function specified by any particular standard COBOL element, then the implementation must not require, except for Environment Division elements, the specification of that external function in place of or in addition to that standard language element:

The following qualifications apply to the American National Standard COBOL

specification:

a. There are certain language elements which pertain to specific types of hardware components. In order for an implementation to meet the requirements of this standard, the implementor must specify the minimum hardware configuration required for that implementation and the hardware components that it supports. Further, when support is thus claimed for a specific hardware component, all standard language elements that pertain to that component must be implemented if the module in which they appear is included in the implementation. Language elements that pertain to specific hardware components for which support is not claimed, need not be implemented. However, the absence of such elements from an implementation of American National Standard COBOL must be specified.

b. An implementation of American National Standard COBOL may include the ENTER statement or not, at the option of the implementor.

c. An implementation that includes, in addition to a specified level of each of the functional processing modules and of the Nucleus, elements or functions that either are not defined in the American National Standard COBOL specification or are defined in a given level of a standard module not otherwise included in the implementation, meets the requirements of this Standard. This is true even though it may imply the extension of the list of reserved words by the implementor, and prevent proper compilation of some programs that meet the requirements of this Standard. The implementor must specify any optional language (language not defined in a specified level but defined elsewhere in the Standard) or extensions (language elements or functions not defined in this Standard) that are included in the implementation.

d. In general, the American National Standard COBOL specification specifies no upper limit on such things as the number of statements in a program, the number of operands permitted in certain statements, etc. It is recognized that these limits will vary from one implementation of American National Standard COBOL to another and may prevent the proper compilation of some programs that meet the requirements of this standard.

IMPLEMENTOR-DEFINED LANGUAGE SPECIFICATIONS

The language elements in the following lists depend on implementor definitions to complete the specificaton of the syntax or rules for the elements.

The elements whose syntax is partly implementor-defined are:

Element	Implementor-Defined Aspect
SOURCE-COMPUTER paragraph	computer-name
OBJECT-COMPUTER paragraph	computer-name
MEMORY SIZE clause	integer
alphabet-name	implementor-name; whether implementor-names are provided.

SPECIAL-NAMES paragraph	implementor-name
ASSIGN clause	implementor-name
VALUE OF clause	implementor-name; whether implementor-names are provided.
RERUN clause	implementor-name and the form; the implementor provides at least one of seven specified forms.
CALL and CANCEL statements	relationship between operand and the referenced program.
COPY statement	relationship between library-name text-name, and the library.
ENTER statement	language-name
Margin R	The location.
Area B	The number of character positions.
Qualification	The number of qualifiers; at least five must be supported.

The elements whose effect is partly implementor-defined are:

Element	Implementor-Defined Aspect
alphabet-name	The correspondence between native and foreign character sets.
implementor-name switches	Whether setting can change during execution.
USAGE IS COMPUTATIONAL clause	Representation and whether automatic alignment occurs.
USAGE IS INDEX clause	Representation and whether automatic alignment occurs.
SYNCHRONIZED clause	Whether implicit FILLER positions are generated; their effect on the size of group items and redefining items.
ACCEPT statement	Maximum size of one transfer of data in Level 1 Nucleus.
DISPLAY statement	Maximum size of one transfer of data in Level 1 Nucleus.
Numeric test	Representation of valid sign in the absence of the SIGN IS SEPARATE

clause.

Comparison of nonnumeric items	Collating sequence, where NATIVE or implementor-name collating sequence is implicitly or explicitly specified.
Arithmetic expressions	Number of places carried for intermediate results.

Elements That Pertain to Specific Hardware Components

The standard language elements in the list that follows pertain to specific types of hardware components. These language elements must be implemented in an implementation of American National Standard COBOL when support is claimed, by the implementor, for the specific types of hardware components to which they pertain, and the module in which they are defined is included in that implementation.

Element	Hardware Component
-SET clause	Device capable of supporting the specified code.
LTIPLE FILE TAPE clause	Reel
CLOSE...REEL/UNIT statement	Reel or mass storage
CLOSE...NO REWIND statement	Reel or mass storage
OPEN...REVERSED statement	Reel with the capability of making records available in the reversed order; mass-storage with the capability of making records available in the reversed order.
OPEN...NO REWIND statement	Reel or mass storage
OPEN...I-O statement (Sequential I-O only)	Mass storage
OPEN EXTEND statement	Reel or mass storage
REWRITE statement (Sequential I-O only)	Mass storage
SEND...BEFORE/AFTER ADVANCING statement	Devices capable of vertical positioning; devices capable of action based on mnemonic-names.
USE...I-O (Sequential I-O only)	Mass storage
WRITE...BEFORE/AFTER	Devices capable of vertical posi-

ADVANCING

tioning; devices capable of action based on mnemonic-name.

1.5 The Federal COBOL Standard

The COBOL compiler validation results enclosed in this document reflect the degree to which the subject COBOL compiler implements the Federal COBOL Standard. The Federal COBOL Standard is essentially the same as the American National Standard Programming Language COBOL, X3.23-1974, with two exceptions:

> The Federal COBOL Standard defines 4 levels and the ANSI Standard defines only the minimum COBOL implementation and the full standard. Low and High levels of the Federal COBOL Standard (see 1.5.1) correspond to the above two ANSI levels (minus the Report Writer module). Two additional levels, low-intermediate and high-intermediate have been included in the Federal Standard between the highest and lowest subsets. These additional levels accommodate hardware which cannot support the full standard, but which is capable of implementing more than the minimum standard.
>
> The Federal COBOL Standard states that the Report Writer Module is not mandatory in any Federal level, but that the specifications contained in X3.23-1974 should be used to the extent practical, consistent with requirements.

The Federal COBOL Standard requires that a compiler contain as a minimum the elements specified in at least one of the Federal levels. No restrictions are imposed on the inclusion of selected features from higher levels or even unique vendor extensions. Compatibility among various implementations of a given level containing additional features must be controlled by management imposed standards and restrictions.

1.5.1 Federal Standard COBOL Levels

a. Federal Standard COBOL specifications are the language specifications contained in American National Standard Programming Language COBOL, X3.23-1974. For purposes of the Federal Standard, the modules defined in X3.23-1974 are combined into four levels. Not all computers are large enough to accommodate a COBOL compiler containing the full ANSI Standard. Therefore, the Federal Government requires that all compilers acquired by its agencies contain as a minimum one of the four Federal levels, depending on machine size, configuration and user needs. The knowledge that all computers will support at least one of these four subsets simplifies the task of developing machine-independent COBOL programs.

b. The four levels of Federal Standard COBOL are identified as: Low, Low-Intermediate, High-Intermediate, and High. Each Federal Standard COBOL level is composed of either the high or low levels of the nucleus and ten of the eleven Functional Processing Modules (FPMs) defined in X3.23-1974. The four Federal Standard COBOL levels are reflected in the following table. The numbers in the table refer to the level within the FPM or nucleus as designated in X3.23-1974, and a dash in the table denotes that the corresponding FPM is omitted.

	Low Level	Low Intermediate Level	High Intermediate Level	High Level
NUCLEUS	1	1	2	2
FPMs				
TABLE HANDLING	1	1	2	2
SEQUENTIAL I-O	1	1	2	2
RELATIVE I-O	-	1	2	2
INDEXED I-O	-	-	-	2
SORT-MERGE	-	-	1	2
REPORT WRITER	-	-	-	-
SEGMENTATION	-	1	1	2
LIBRARY	-	1	1	2
DEBUG	-	1	2	2
INTER-PROGRAM COMMUNICATION	-	1	2	2
COMMUNICATION	-	-	2	2

1.5.2 Conformance to Federal Standard COBOL

A compiler implemented in conformance to Federal Standard COBOL must meet at least the following requirements.

a. The implementation must include all of the language elements of at least one of the levels of Federal Standard COBOL.

b. The implementation must meet all of the requirements defined in American National Standard COBOL, X3.23-1974, Section I, paragraph 1.5, Definition of An Implementation of American National Standard COBOL which is provided in section 1.4 of this VSR.

c. The implementation must provide a facility for the user to optionally specify a level of Federal Standard COBOL for monitoring his source program at compile time. The monitoring will be an analysis of the syntax used in a source program against the syntax included in the specified level of Federal Standard COBOL. Any syntax used in the source program that does not conform to that allowed by the user selected level of Federal Standard COBOL will be diagnosed. The syntax diagnosed as not conforming to the specified level will be identified to the user through a diagnostic message on the source program listing. The diagnostic message will contain, at least: (1) The identification of the source program line number in which the nonconforming syntax occurs, (2) the identification of the level of Federal Standard COBOL that supports the syntax or that the syntax is nonstandard COBOL.

1.6. Use of the VSR

The Federal COBOL Compiler Testing Service may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation are only for the purpose of satisfying United States Government requirements, and apply only to the computer system, operating system release, and compiler version identified in the VSR.

The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

1.7 Sources of Additional Information

FIPS PUB 21-1 defines the Federal COBOL Language Standard. This publication is available from the Office of ADP Standards Management, National Bureau of Standards, Washington, D. C., 20234.

The detailed COBOL language specifications are given in the publication "American National Standard Programming Language COBOL, X3.23-1974", available from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

An explanation of the COBOL Compiler Validation System is contained in the CCVS User's Guide. This document explains how to run the compiler validation system. The User's Guide and a magnetic tape containing a copy of the CCVS programs are available from the National Technical Information Service, Springfield, Virginia, 22151. (Ordering information can be obtained from the Federal COBOL Compiler Testing Service.)

1.8. Requests for Interpretation

Questions regarding this VSR or the CCVS in general should be forwarded to the FCCTS. If any problem cannot be adequately resolved through the FCCTS, the request for interpretation will be forwarded to the Federal COBOL Interpretation Committee for final resolution.

A brochure describing the validation process including the procedures for requesting a validation and resolution of questions involving interpretation of the current Federal Standard is available from the Department of the Navy, Federal COBOL Compiler Testing Service, Washington, D.C. 20376.

1.9 Modules and Language Elements Excluded from Testing

During an official validation, certain CCVS tests may not be used, and certain facilities provided by the subject compiler may not be tested.

1.9.1 Federal Standard COBOL Approved Interpretations

The National Bureau of Standards published in the Federal Register Vol. 41 No. 179, September 14, 1976, an approved interpretation of Federal Standard COBOL as pertains to the evaluation of arithmetic expressions in the COMPUTE statements. This interpretation states that "size of the intermediate result field is implementor-defined."

Since the results of evaluating arithmetic expressions are not predictable, all COMPUTE statements and IF statements containing arithmetic expressions have been removed from the COBOL Compiler Validation System.

1.9.2 Report Writer Module

FIPS PUB 21-1 excludes the Report Writer Module from the Federal COBOL Standard. However, the Report Writer Module is still tested during a validation if support for that module is claimed by the compiler vendor.

1.9.3 Communication Module

Although it is part of Federal Standard COBOL as defined by FIPS PUB 21-1, the Communication Module is not currently tested in the course of an official validation for two specific reasons. First, a large volume of requests for interpretation on this module have been submitted to the cognizant ANSI committee (X3J4) for resolution. Secondly, facilities for testing were insufficient to determine the validity of the Communication Module test programs during the development of CCVS74.

1.9.4 Vendor Omissions or Extensions

Language elements are not tested which have been legitimately omitted from the implementation by the implementor (refer to 1.4). Additionally, no implementor extensions to the standard COBOL language are tested in any way.

1.10 Timeliness of the Validation Summary Reports

The timeliness of the Validation Summary Report is important. Compilers and their related operating system software are modified several times a year. The Compiler Validation System used to validate compilers is also updated during the life of the system. Therefore to ensure that the latest version of both the vendor's compiler and the Validation System are the latest officially released versions, check with the:

Director
Federal COBOL Compiler Testing Service
Department of the Navy
Washington, D. C. 20376
(202) 697-1247

Please use the Validation Summary Report number of this report when corresponding with the Testing Service.

SECTION 2. DETAILED EVALUATION OF ERRORS.

This section summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System (CCVS). The version of the CCVS used during this validation is shown inside the front cover of the VSR.

Section 2 is made up of a variable number of subsections. The number of subsections is dependent on the level of Federal COBOL being validated. There will be a subsection for each level of each module which is validated. If the high level of a module is validated then there will be two subsections for that module; one for the low level and one for the high level.

A validation of the low level of Federal Standard COBOL would result in three subsections being present. One for Nucleus level 1, one for Sequential I-O level 1, and one for Table Handling level 1.

Each error or deviation noted in this section makes reference to a program or functional COBOL module contained in Appendix A (Validation Summary Working Document). This reference provides the documented results of an occurrence of errors/deviataions detected during the running of the CCVS using the compiler within the environment identified within this document. The Validation Summary Working Document is presented in sequence by functional module, functional module level and program number as defined below.

Each program in the COBOL Compiler Validation System is identified by a 5-character program name. The name associates the routine with the functional processing module and level of American National Standard Programming Language COBOL tested within the program.

The five character name has the general format XXNMM". The first two characters are alphabetic and identify the functional module tested by the program. The permissable values are:

NC - Nucleus
TH - Table Handling
SQ - Sequential I-O
RL - Relative I-O
IX - Indexed I-O
ST - Sort-Merge
RW - Report Writer
SG - Segmentation
LB - Library
DB - Debug
IC - Inter-Program Communication
CM - Communication

The third character of the audit routine name is either a 1 or 2, and identifies the level of the functional module being tested. Each module and level is represented by several programs. The fourth and fifth characters of the program name are sequence numbers for programs which test features in the same level of the same functional processing module.

As an example, the program name NC210 is the tenth program in the series of

routines which test the second level of the Nucleus module.

Description of Section 2.

Each error/deviation is noted by number in the left hand margin opposite the language element in question. This number is used in section 3 to categorize errors by Federal level (See 1.5.1). Inserted directly below the language element is a brief description of the error. To the right of the langauge element is a page reference to X3.23-1974, American National Standard Programming Language COBOL. The reference at the end of the description of the error is to Appendix A which contains the detailed information collected during the validation. The reference is made up of the routine name followed by an A or B (A for compile time or syntax error and B for execution time or semantic error) and a number which makes the error unique in Appendix A.

Example:

2.1 Nucleus Level 1

.
.
.

Operational symbols: S V P II-21

2.1.9 ----------

* The scaling character 'P' is not permitted in a
* PICTURE character-string.
* (NC101.A.2)

.
.
.

2.2 Sequential I-O Level 1

2.1.9 represents the ninth error for Nucleus Level 1

II-21 represents the page in X3.23-1974 where the language element is defined

* Boxes the description of the error/deviation

NC101.A.2 represents:

Program name - NC101
Syntax error - A
second error - 2

2.1 NUCLEUS LEVEL 1

2.2 NUCLEUS LEVEL 2

All elements of 1 NUC 1,2 are a part of 2 NUC 1,2

Language Concepts. I-75
- Characters used for punctuation. I-65
  - , comma
  - ; semicolon
- Characters used for arithmetic operations. I-52
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - ** exponentiation
- Characters used in relations I-66
  - = equal to
  - > greater than
  - < less than
- Separators I-75
  - The separators, semicolon and comma, are allowed . . II-1
- Character-strings. I-76
- COBOL words. I-76
  - User-defined words I-76
    - condition-name
    - data-name
      - Need not begin with an alphabetic character. . II-1
      - May be qualified if necessary for uniqueness . II-1
  - Reserved words I-79
    - Figurative constants I-80
      - ZEROS; ZEROES
      - SPACES
      - HIGH-VALUES
      - LOW-VALUES
      - QUOTES
      - ALL literal
    - Connectives I-79
      - Qualifier connectives: OF, IN
      - Series connectives: , (separator comma) and ; (separator semicolon)
      - Logical connectives: AND, OR, AND NOT, OR NOT
- Qualification I-87

Reference format I-105
- Continuation of lines (continuation of words and numeric literals is allowed) II-1

Identification Division. I-94
- The DATE-COMPILED paragraph. II-4

Environment Division
- The SPECIAL-NAMES paragraph. II-8
  - alphabet-name clause
    - literal

Data Division. I-97

2.3 TABLE HANDLING LEVEL 1

2.4 TABLE HANDLING LEVEL 2

All elements of 1 IBL 1,2 are a part of 2 IBL 1,2

Data Division
- The OCCURS clause III-2
  - integer-1 TO integer-2 DEPENDING ON data-name
  - ASCENDING/DESCENDING data-name
    - data-name series
  - ASCENDING/DESCENDING series

Procedure Division
- The SEARCH statement. III-7
  - VARYING phrase
  - AT END phrase
  - WHEN phrase
- The SEARCH ALL statement. III-7
  - AT END phrase
  - WHEN phrase

2.5 SEQUENTIAL I-O LEVEL 1

2.5.1 ---

*A WRITE ... BEFORE ADVANCING ZERO statement does not execute
*correctly. The system prints the line and then advances to
*a new page. (SQ101 B)

2.6 SEQUENTIAL I-O LEVEL 2

All elements of 1 SEQ 1,2 are a part of 2 SEQ 1,2

Language Concepts
- Special register I-80
  - LINAGE-COUNTER. IV-3

Environment Division
- The FILE-CONTROL paragraph IV-4
- The file control entry IV-4
  - SELECT clause
    - OPTIONAL phrase
  - RESERVE integer AREA(S) clause
- The I-O-CONTROL paragraph. IV-6
  - SAME RECORD AREA clause
  - SAME RECORD AREA series
  - MULTIPLE FILE TAPE clause

Data Division
- The file description entry. IV-10
- The BLOCK CONTAINS clause IV-11
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The LINAGE clause IV-15

2.6.1 ----------
*Programs containing the LINAGE clause did not execute correctly.
* (SQ202 B, SQ213 B, SQ214 B, SQ215 B)

  - FOOTING phrase
  - TOP phrase
  - BOTTOM phrase
- The VALUE OF clause IV-19
  - implementor-name IS data-name
  - implementor-name IS data-name series

Procedure Division
- The CLOSE statement IV-20
  - NO REWIND, REMOVAL, or LOCK
  - file-name series
- The OPEN statement. IV-24
  - INPUT
    - REVERSED

2.6.2 ----------
*OPEN ... REVERSED is not supported by this system. This is not an
*error as OPEN ... REVERSED is a language element that pertains to
*a specific type of hardware component. See section 1.4.
* (SQ206 A)

    - NO REWIND
  - OUTPUT
    - NO REWIND
  - EXTEND
  - file-name series
  - INPUT, OUTPUT, I-O, and EXTEND series

2.6.3 ----------

*WRITE statements in programs containing the LINAGE clause did not
*execute correctly. (SQ202 B, SQ213 B, SQ214 B, SQ215 B)

BEFORE/AFTER identifier LINES
BEFORE/AFTER mnemonic-name
AT END-OF-PAGE imperative-statement

2.6.4 ----------

*An END-OF-PAGE clause with a WRITE statement executed incorrectly.
*The printed line followed the last detail line, but it should have
*been the first line on a new page. (SQ213 B)

2.7 RELATIVE I-O LEVEL 1

2.8 RELATIVE I-O LEVEL 2

All elements of 1 REL 0,2 are a part of 2 REL 0,2

2.9 INDEXED I-O LEVEL 1

2.10 INDEXED I-O LEVEL 2

All elements of 1 INX 0,2 are a part of 2 INX 0,2

Environment Division
- The FILE-CONTROL paragraph VI-5
- The file control entry VI-5
  - SELECT clause
  - RESERVE integer AREA(S) clause
  - ACCESS MODE IS DYNAMIC clause
  - ALTERNATE RECORD KEY clause

2.10.1 ---

*Alternate keys are not implemented on this system.
* (IX205 A, IX206 A, IX207 A, IX208 A)

    - WITH DUPLICATES phrase
- The I-O-CONTROL paragraph. VI-8
  - SAME RECORD clause
  - SAME RECORD AREA series

Data Division
- The file description entry VI-12
- The BLOCK CONTAINS clause. VI-13
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause. VI-17
  - implementor-name IS data-name
  - implementor-name IS data-name series

Procedure Division
- The READ statement VI-24
  - KEY IS phrase
  - NEXT RECORD
- The START statement. VI-30
  - KEY IS phrase
  - INVALID KEY phrase
- The USE statement. VI-32
  - EXCEPTION/ERROR PROCEDURE
    - ON file-name series

2.11 SORT-MERGE LEVEL 1

Language Concepts
- User-defined words. I-76
  - file-name

Environment Division
- The FILE-CONTROL paragraph. VII-2
- The file control entry. VII-2
  - SELECT clause
  - ASSIGN TO implementor-name clause

Data Division
- File Section. VII-5
- The sort-merge file description entry VII-5
- The DATA RECORDS clause VII-6
- The RECORD CONTAINS clause. VII-7

Procedure Division
- The RELEASE statement VII-12
  - FROM phrase
- The RETURN statement. VII-13
  - INTO phrase
  - AT END phrase
- The SORT statement (only one SORT statement, a STOP RUN statement, and any associated input-output procedures allowed in the nondeclarative portion of a program) VII-14
  - KEY data-name
    - data-name series
  - ASCENDING series
  - DESCENDING series
  - mixed ASCENDING/DESCENDING
  - INPUT PROCEDURE phrase
    - THRU
  - USING phrase
  - OUTPUT PROCEDURE phrase
    - THRU
  - GIVING phrase

2.12 SORT-MERGE LEVEL 2

All elements of 1 SRT 0.2 are a part of 2 SRT 0.2

Environment Division
- The FILE-CONTROL paragraph. VII-2
- The file control entry. VII-2
  - SELECT clause
- The I-O-CONTROL paragraph VII-3
  - SAME RECORD AREA clause
  - SAME SORT/SORT-MERGE AREA clause
  - SAME series

Procedure Division
- The MERGE statement VII-8
  - KEY data-name
    - data-name series
  - ASCENDING series
  - DESCENDING series
  - mixed ASCENDING/DESCENDING
  - COLLATING SEQUENCE phrase
  - USING phrase
  - OUTPUT PROCEDURE phrase
    - THRU
  - GIVING phrase
- The SORT statement (multiple SORT statements are permitted). VII-14
  - COLLATING SEQUENCE phrase

2.13 REPORT WRITER LEVEL 1

2.13.1 ---

*The report writer module is not implemented on this system.

Language Concept
- User-defined words I-76
  - file-name
  - report-name
- Special registers I-80
  - LINE-COUNTER VIII-1
  - PAGE-COUNTER VIII-1

Data Division
- Report Section VIII-2
- The file description entry VIII-3
- The report description entry VIII-4
- The report group description entry VIII-6
- The BLOCK CONTAINS clause VIII-24
- The CODE clause VIII-25
- The CODE-SET clause VIII-26
- The COLUMN NUMBER clause VIII-27
- The CONTROL clause VIII-28
  - data-name
  - data-name series
  - FINAL
  - FINAL data-name series
- The data-name clause VIII-30
- The GROUP INDICATE clause VIII-31
- The LABEL RECORDS clause VIII-32
- The LINE NUMBER clause VIII-33
  - integer
    - NEXT PAGE
  - PLUS integer
- The NEXT GROUP clause VIII-35
  - integer
  - PLUS integer
  - NEXT PAGE
- The PAGE clause VIII-36
  - integer LINES
  - HEADING
  - FIRST DETAIL
  - LAST DETAIL
  - FOOTING
- The PICTURE clause II-18
- The RECORD CONTAINS clause VIII-39
- The REPORT clause VIII-40
  - report-name series
- The SOURCE clause VIII-41
- The SUM clause VIII-42
  - UPON data-name series
  - RESET phrase
- The TYPE clause VIII-45
  - REPORT HEADING (RH)

2.14 SEGMENTATION LEVEL 1

Language Concepts
- User-defined words. I-76
  - segment-number

Procedure Division
- Segment-numbers Ix-4
  - Fixed segment-number range 0 through 49
  - Non-fixed segment-number range 50 through 99
  - All sections with the same segment-number must be together in the source program

2.15 SEGMENTATION LEVEL 2

All elements of 1 SEG 0,2 are a part of 2 SEG 0,2

Environment Division
The OBJECT-COMPUTER paragraph
SEGMENT-LIMIT. IX-5

Procedure Division
Segment-numbers IX-4
Sections with the same segment-number need not
be physically contiguous in the source program

2.16 LIBRARY LEVEL 1

2.17 LIBRARY LEVEL 2

All elements of 1 LIB 0,2 are a part of 2 LIB 0,2

2.17.1 ----------

*Level 2 of the Library module is not implemented on this system.

Language Concepts
User-defined words I-76
library-name

All divisions
The COPY statement X-2
OF library-name
REPLACING phrase

2.18 DEBUG LEVEL 1

2.18.1 ----------

*Level 1 of the Debug module is not implemented on this system.

Language Concepts
Special registers. I-80
DEBUG-ITEM. XI-1

Environment Division
The SOURCE-COMPUTER paragraph
WITH DEBUGGING MODE clause. XI-3

Procedure Division
USE FOR DEBUGGING statement. XI-4
procedure-name
procedure-name series
ALL PROCEDURES
Debugging lines. XI-10

2.19 DEBUG LEVEL 2

2.19.1 ----------

*Level 2 of the Debug module is not implemented on this system.

All elements of 1 DEB 0,2 are a part of 2 DEB 0,2

Procedure Division
USE FOR DEBUGGING statement. XI-4
ALL REFERENCES OF identifier series
file-name series
cd-name series

2.20 INTER-PROGRAM COMMUNICATIONS LEVEL 1

2.21 INTER-PROGRAM COMMUNICATIONS LEVEL 2

All elements of 1 IPC 0,2 are a part of 2 IPC 0,2

Procedure Division
- The CALL statement. XII-5
  - identifier
  - ON OVERFLOW phrase
- The CANCEL statement. XII-7

2.22 COMMUNICATION LEVEL 1

---

* The Communication Module is not currently evaluated as part of
* an official validation. See Section 1 Paragraph 1.9.3.

Language Concepts
- User-defined words. I-76
  - cd-name

Data Division
- Communication Section XIII-2
- The communication description entry XIII-3
  - FOR INPUT clause
    - END KEY
    - MESSAGE COUNT
    - MESSAGE DATE
    - MESSAGE TIME
    - SYMBOLIC QUEUE
    - SYMBOLIC SOURCE
    - SYMBOLIC SUB-QUEUE-n
    - STATUS KEY
    - TEXT LENGTH
  - FOR OUTPUT clause
    - DESTINATION COUNT
    - DESTINATION TABLE
      - INDEXED BY
    - ERROR KEY
    - SYMBOLIC DESTINATION
    - STATUS KEY
    - TEXT LENGTH

Procedure Division
- The ACCEPT MESSAGE COUNT statement. XIII-12
- The DISABLE statement XIII-13
  - INPUT
  - OUTPUT
  - KEY identifier/literal
- The ENABLE statement. XIII-15
  - INPUT
  - OUTPUT
  - KEY identifier/literal
- The RECEIVE statement XIII-17
  - MESSAGE
  - INTO identifier
  - NO DATA phrase
- The SEND statement. XIII-20
  - FROM identifier-1 WITH
  - WITH EMI
  - WITH EGI
  - BEFORE/AFTER ADVANCING
    - identifier-3 LINES
    - integer LINES
    - mnemonic-name

PAGE

2.23 COMMUNICATION LEVEL 2

---

* The Communnication Module is not currently evaluated as part of
* an official validation. See Section 1 Paragraph 1.9.3.

All elements of 1 COM 0,2 are a part of 2 COM 0,2

Communication Section
The communication description entry. XIII-3
FOR INPUT
INITIAL

Procedure Division
The DISABLE statement. XIII-13
INPUT
TERMINAL
The ENABLE statement XIII-15
INPUT
TERMINAL
The RECEIVE statement. XIII-17
SEGMENT
The SEND statement XIII-20
FROM identifier-1
WITH identifier-2
WITH ESI

SECTION 3. COMPILER STATUS

3.1 Federal Standard COBOL

Section 1.5 explains the four levels of Federal Standard COBOL and their relation to American National Standard COBOL. This section lists the discrepancies described in Section 2 by the Federal level in which the problem occurs. All errors listed for a lower level are also errors in any higher level, even though they are listed only in the lower level. The paragraph number from Section 2 is used to reference the errors in each Federal level.

3.1.1 Low Level

2.5.1 WRITE...BEFORE ADVANCING ZERO statement.

3.1.2 Low-Intermediate Level

2.19.1 Debug module level 1 is not implemented.

3.1.3 High-Intermediate

2.6.1 LINAGE clause - execution results not correct.
2.6.3 WRITE statements in programs containing LINAGE clause.
2.6.4 WRITE...AT END-OF-PAGE statement.
2.19.1 Debug module level 2 is not implemented.

3.1.4 High Level

2.10.1 Alternate keys are not implemented.
2.17.1 Level 2 of the Library module is not implemented.

3.2 American National Standard COBOL

Full American National Standard COBOL consists of the entire set of language elements defined in the ANSI COBOL standard (refer to 1.7). It is also the equivalent of high level Federal Standard COBOL plus the Report Writer module. Therefore, this section lists only those discrepancies found while validating the Report Writer Module.

2.13.1 Report Writer module is not implemented.

SECTION 4. SOFTWARE ENVIRONMENT.

The compiler referenced in this document was validated using the software environment described in this section. When using a modification of the described environment, the compiler may or may not continue to conform to the Standard. It should be noted that during the validation process, an attempt is made to validate as many different options as possible.

The use of compiler options, implementor-names in the Environment Division and any form of optimization which is not described in this report could cause the compiler to produce a program that does not perform according to the specifications of Standard COBOL. Only the environment described in this document has been used with this compiler to satisfy the requirements of FIPS PUB 21-1 and FPMR 101-32.1305.1a. (Any deviations which must be corrected as per the referenced FPMR are described in Sections 2 and 3 of this report.)

1. Options or parameters used on the processor call statement for the compiler: The following options/parameters were used during the validation.

Several different options were used during this validation. The options which were used are:

sc - source
map - map, cross reference, source

Some of the routines required special options for execution:

The cs (Cobol_switch) option was required to set the switch settings for NC103 and NC211.

The tu (continue) option was required to override the SIZE ERROR condition in TH210 and ST202.

The sd (sort directory) option was required for ST116.

The nep (no end page) option was required when printing the output reports for SQ101, SQ201, SQ202, SQ212 through SQ215, to override the default page definition.

2. Environment Division implementor-names.

Printer destined files

REPORT-PRINTER

Tape files

X01-TAPE CATALOG NAME IS "X101"

Sequential Mass-storage files

X14-VIRTUAL

Random Access files

X21-VIRTUAL

Sort files (SD)

X27-VIRTUAL

Switch names

SWITCH-n

Source Computer names

MULTICS

Object Computer names

MULTICS

3. Optimization. The compiler may or may not have optimization features. If there was an optimization feature available, it was used during the validation process (during a separate execution of the Compiler Validation System) to determine if its use causes the compiler to produce a program which does not give the expected results. If the optimization is invoked through the compiler call statement then it is mentioned in paragraph 1 above. If it is invoked through the introduction of syntax in other than the Data and Procedure Divisions of the source program it is shown below. Optimization which would require modification to the Data and Procedure Divisions is not considered in this report in that it is beyond the scope of the use of standard COBOL and the validation process.

There is no specific optimization feature for this compiler.

4. Compiler.

Multics COBOL Version 2.3

5. Operating system.

Multics Release MR4.0

SECTION 5. ASCII VALIDATION

5.1 Purpose of ASCII Validation

The ASCII Validation is performed by running a sequence of three CCVS74 programs (SQ118, SQ119, SQ120) using special procedures. The purpose of this special run is to validate that the compiler/operating system being tested is capable of processing ASCII code represented on magnetic tape and punched cards that was produced (in accordance with the appropriate American National Standard) by another system. There is also a magnetic tape and a card file created during the validation which will be taken to another system for further processing. The purpose is to determine whether the compiler/operating system being tested can also produce ASCII represented on magnetic tape and punched cards which can be processed by a another computer system.

5.2 Applicable ANSI Standards

The ASCII Validation is based on several American National Standards and presumes their support by the compiler/operating system being validated. These are:

1. American National Standard Programming Language COBOL X3.23-1974

   - The CODE-SET clause is used to read and write the ASCII files.

   - The PROGRAM COLLATING SEQUENCE clause is used to process the data in ASCII mode as well as native mode.

   - The SIGN...SEPARATE clause is used for signed data and all data is in the DISPLAY (character) mode.

2. American National Standard Code for Information Interchange (ASCII) X3.4-1968. (Note that this describes the code, not the labeling and tape recording formats.)

3. American National Standard Hollerith Punched Card Code, X3.26-1970.

4. American National Standard Magnetic Tape Labels for Information Interchange, X3.27-1969.

5. American National Standard Recorded Magnetic Tape for Information Interchange (800 CPI, NRZI), X3.22-1967.

6. American National Standard Recorded Magnetic Tape for Information Interchange (1600 CPI, PE), X3.39-1973.

The language of the 1974 COBOL Standard provides the capability to accept, process, and produce ASCII code. The ASCII Standard describes the code insofar as the bit arrangement and configuration, but does not address recording techniques, record formats or any labeling scheme. The 800 CPI, NZRI magnetic tape recording standard was used to establish the recording density and techniques. (1600 CPI, PE based on X3.39-1973 "Recorded Magnetic Tape for Information Inter-

change" could be used under special arrangements.) The tape labeling scheme used in these tests is based on X3.27-1969 but is also compatible with the revision to that tape label standard. Only the VOL1, HDR1, and EOF1 labels are used. The records are fixed length and unblocked.

5.3 ASCII Validation Process

During the validation, the Validation Manager for the Federal COBOL Compiler Testing Service uses the ASCII-encoded magnetic tape and card files in addition to the normal tape files associated with a validation. For the ASCII portion of the validation the following steps are performed:

1. The tape file and card deck (produced on another computer system) are used as input to several programs designed to validate whether the system being validated can accept and process the data as defined by the respective standards. Any changes made during this validation to the source programs reading the data are noted below in 5.4.1.

2. A tape file and card file are produced during the validation which should prove to be identical to the files described in 1 above. These two files are then processed on a different computer system to determine the degree to which the system being validated supports the ASCII standard. Any changes made during this validation to the source program producing the data are noted below in 5.4.2.

5.4 Results for This Validation

1. The Multics system processed the card deck, the ANSI labeled tape, and the unlabeled tape correctly.

2. The Multics system produced an ANSI labeled tape and a card deck which were both verified later as being correct in format and code set. The data records on the tape were preceded by VOL1, HDR1 and HDR2 label records and a tape mark; and were followed by a tape mark, EOF1 and EOF2 label records, and a double tape mark. All labels were correct according to the applicable ANSI Standard for magnetic tape labels.

APPENDIX A

VALIDATION SUMMARY WORKING DOCUMENT

A-1 This appendix is a working paper produced during the validation and documents the results of the compilation and execution of each of the programs comprising the CCVS. The results contained herein are based on the use of the compiler within the Validation Environment identified in this appendix. This appendix (Validation Summary Working Document) is not part of the official Validation Summary Report (VSR) and is not intended to reflect in any way the compiler's usefulness or degree of conformance to the language specifications.

The reader of this appendix should keep in mind that the same problem area may appear in more than one program, but is considered only as one single discrepancy and as such is reflected only once in the body of the VSR. (The VSR will in turn only reference the first occurrence of the problem in the appendix.)

The reference document for COBOL is FIPS PUB 21-1 (X3.23-1974).

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION:	Multics COBOL Version 2.3
COMPUTER SYSTEM:	H6180 (Multics)
OPERATING SYSTEM:	Multics Release MR4.0

COMMUNICATION LEVEL 1 and LEVEL 2

The Communication programs were not run.

DEBUG LEVEL 1 and LEVEL 2

The Debug module is not implemented for this compiler and the Debug programs were not run.

INTER-PROGRAM COMMUNICATION LEVEL 1

IC101 through IC115, IC151 and IC152

A. Compilation

No errors.

B. Execution

No errors.

INTER-PROGRAM COMMUNICATION LEVEL 2

IC201 through IC208

A. Compilation

No errors.

B. Execution

No errors.

INDEXED I-O LEVEL 1

IX101 through IX107

A. Compilation

No errors.

B. Execution

No errors.

INDEXED I-O LEVEL 2

IX201 through IX204

A. Compilation

No errors.

B. Execution

No errors.

IX205 through IX208

A. Compilation

Alternate keys are not implemented for this compiler. Statements which referenced alternate keys caused fatal diagnostic messages.

B. Execution

The tests referencing alternate keys had to be deleted. The other tests in these programs executed correctly.

LIBRARY LEVEL 1

LB101 through LB107

A. Compilation

No errors.

B. Execution

No errors.

LIBRARY LEVEL 2

Level 2 of the Library Module is not implemented for this compiler and the level 2 Library programs were not run.

NUCLEUS LEVEL 1

NC101 through NC120, and NC151 through NC165

A. Compilation

No errors.

B. Execution

No errors.

NUCLEUS LEVEL 2

NC201 through NC218

A. Compilation

No errors.

B. Execution

No errors.

RELATIVE I-O LEVEL 1

RL101 through RL109 and RL151 through RL153

A. Compilation

No errors.

B. Execution

No errors.

RELATIVE I-O LEVEL 2

RL201 through RL205

A. Compilation

No errors.

B. Execution

No errors.

REPORT WRITER MODULE

The Report Writer module is not implemented on this system and the Report Writer programs were not run.

SEGMENTATION LEVEL 1

SG101 through SG106

A. Compilation

No errors.

B. Execution

No errors.

SEGMENTATION LEVEL 2

SG201 through SG204

A. Compilation

No errors.

B. Execution

No errors.

SEQUENTIAL I-O LEVEL 1

SQ101

A. Compilation

No errors.

B. Execution

WRT-TEST-31 failed. This test is a WRITE ... BEFORE ADVANCING ZERO statement. The system prints the line and then advances to a new page.

SQ102 through SQ121

A. Compilation

No errors.

B. Execution

No errors.

SQ151

A. Compilation

No errors.

B. Execution

WRT-TEST-31 failed. This test is a WRITE ... BEFORE ADVANCING ZERO statement. The system prints the line and then advances to a new page. (See SQ101.)

SQ152 and SQ153

A. Compilation

No errors.

B. Execution

No errors.

SEQUENTIAL I-O LEVEL 2

SQ201

A. Compilation

No errors.

B. Execution

No errors.

SQ202

A. Compilation

No errors.

B. Execution

This program contains the following LINAGE clause:

```
LINAGE IS 50 LINES
    WITH FOOTING AT 45
    LINES AT TOP 10
    LINES AT BOTTOM 6.
```

WRT-TEST-07 contains a WRITE ... AFTER ADVANCING PAGE statement. The printed line appeared as the last line on a logical page instead of the first line of a new logical page.

The comment lines for WRT-TEST-08 through WRT-TEST-11 were not printed correctly.

SQ203 through SQ205

A. Compilation

No errors.

B. Execution

No errors.

SQ206

A. Compilation

OPEN ... REVERSED is not supported by this system. The *OPT4 M CCVS control card was used to delete all OPEN ... REVERSED statements from the source program.

B. Execution

No errors.

S0207 through S0212

A. Compilation

No errors.

B. Execution

No errors.

S0213

A. Compilation

No errors.

B. Execution

This program contains the following LINAGE clause:

```
LINAGE data-name
   FOOTING data-name
   TOP data-name
   BOTTOM data-name.
```

There were many errors in the output report. The blank line following the first logical page was omitted for WRT-TEST-01. The first logical page was also incorrect for WRT-TEST-02 and WRT-TEST-03. A blank line was missing preceeding the detail lines and was added following the detail lines.

WRT-TEST-04 is missing two blank lines following the detail lines. WRT-TEST-05 drops the footing line and the blank line which is supposed to follow the detail line.

The line printed from an EOP clause for WRT-TEST-07 appeared immediately following the last detail line. It should have been the first line on the next page.

Many of the print lines were skewed to the left on the printed output.

S0214

A. Compilation

No errors.

B. Execution

This program contains the following LINAGE clause:

LINAGE 40
TOP 2.

WRT-TEST-01, WRT-TEST-02 and WRT-TEST-03 all contained errors. An extra line was printed at the bottom of each logical page instead of at the top of a new page.

SQ215

A. Compilation

No errors.

B. Execution

The comment line for LIN-TEST-01 was not printed correctly. A single line was printed on two lines.

SQ216 through SQ218

A. Compilation

No errors.

B. Execution

No errors.

SORT-MERGE LEVEL 1

ST101 through ST117

A. Compilation

No errors.

B. Execution

No errors.

SORT-MERGE LEVEL 2

ST201 through ST215

A. Compilation

No errors.

B. Execution

No errors.

TABLE HANDLING LEVEL 1

TH101 through TH111, and TH151 and TH152

A. Compilation

No errors.

B. Execution

No errors.

TABLE HANDLING LEVEL 2

TH201 through TH220

A. Compilation

No errors.

B. Execution

No errors.